A Division of
THE MAIL MART
Box 11102
San Francisco
CA 94101

T-PAL T.M.

PROGRAMMING AMATEUR'S LETTER

Published Monthly
for TRS-80 Owners
"All the Latest
Wrinkles"

THE "DO-IT-YOURSELF" SOFTWARE NEWSLETTER

SAMPLE ISSUE 1979

HALF SIZE

ED V. THORNE EDITOR/PUBLISHER

Welcome to T-PAL, Programming Amateur's Letter, THE "DO-IT-YOURSELF" SOFTWARE NEWSLETTER. That's quite a title and, although it is lengthy, it does describe what we're up to here in San Francisco. Publishing the only software newsletter that concentrates on how to do it YOUR WAY.

This is a sample issue to give you an idea of how we approach things. There are some basic ideas and methods in the following pages, some for Level I and Level II — some for Level II only. Surveys indicate that the great majority of people have either acquired their Level II or plan to do so as soon as possible. Many of these people, having obtained their Level II, are not using the full potential of it either due to a lack of understanding — or overlooking — some of the important features and capabilities of the level II. If you are one of these people, this newsletter is made-to-order for you since we will be explaining and demonstrating ways to use these important features and capabilities.

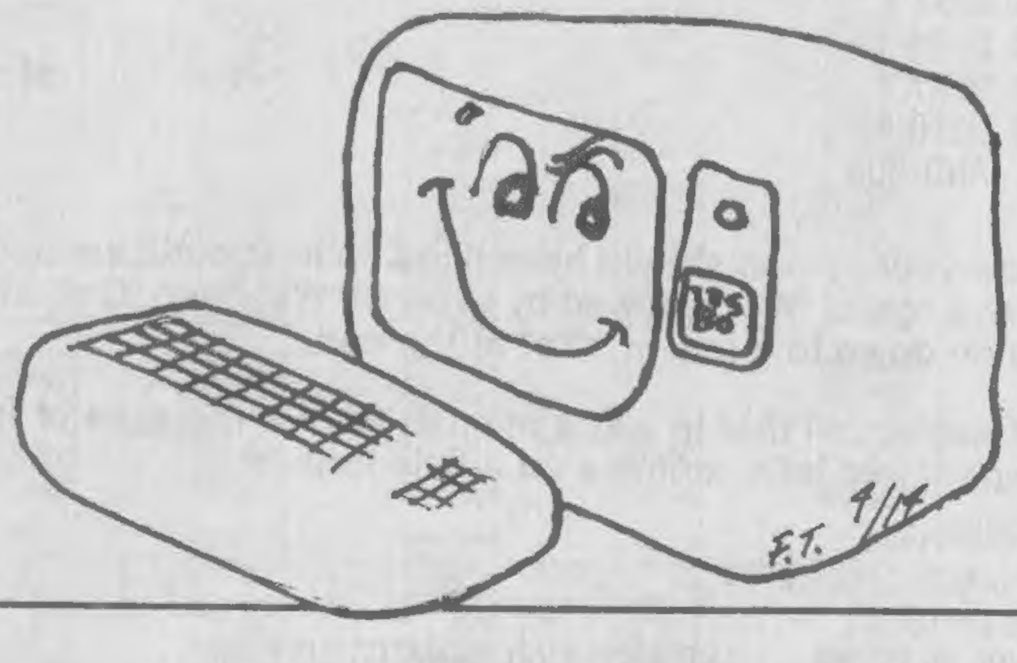
The Diabolical Computer

Speaking of Level I . . . some of those Level I machines were down-right diabolical. We had one of those. It would bide its time, just wait and wait (once I thought I heard it snicker) until we had typed in a program as long as your arm and leg, then it would pounce! I forget just what it was that it would wait for us to do. List or run or something like that. Anyway, WHOP! It would turn the whole program into garbage — unintelligible garbage. It would still list, as I recall, but letters, graphics symbols, numbers, and punctuation were impossibly and hopelessy jumbled. It reminded me of the time the pressure cooker full of spaghetti blew its top at my mother's house. It wasn't funny.If you Level I users ever have that happen to you, don't take it lying down. Put the thing in a strait jacket and rush it down to your Radio Shack dealer for a chip transplant.

Another thing our computer did at the beginning was to get tired. We discovered this one day when we were trying out the termite program from the Level I manual. The first thing this program does (in its own sweet time) is whiten the screen. Paints it white line by line. Well, when the screen was about half white, it started to get shaky. And the whiter it got the shakier it got. By the time the screen was all white and the termites started eating away, it was rolling around like a drunken sailor. We had to put that video unit in a strait jacket too. We began wondering if we should start an ambulance service for sick computers.

But not to knock our good old TRS-80. One of the best things about the whole proposition of owning one is that there are no gray hairs, no six-month wait to have your machine returned, no packaging and mailing. Just take it down to your dealer, he takes it to the rehabilitation center, they give it the cure, and presto! In a day or so your reformed and humbled computer is back at your house working away to make up for lost time

Well, so much for Level I. We graduated from that as quickly as possible when we learned what we could do with Level II. Folks, if you haven't done it yet and are on the fence about the matter, go and take advantage of one of the best buys in the computer world. Get your Level II. It's fantastic.



Before going further I should introduce myself. My name is Ed Thorne and I have spent the better part of my lifetime at sea working on ships. I've done everything on ships from "messman",waiting on tables (a long time ago), to serving as Master (Captain).

While working as a navigator aboard a ship I programmed all of our navigational routines into a programmable calculator and used this daily for routine navigational work. Later I got into using a TRS-80 and adapted the navigational routines for computer use. I was never happy with the programming help I was able to find in books and manuals and eventually started contacting others who showed a talent for programming.

In 1977 I organized the Mail Mart of San Francisco, a company that publishes and distributes information — mostly by mail. Still dissatisfied with the programming assistance available, I decided to set up my own service for beginning programmers in a form that I would have liked to have had available. I have sought out people who have an

(Countinued on page 2)

Introduction *(Continued from page 1)*

exceptional talent, genius if you will, for programming the TRS-80. Working with these people has been a wonderful and enriching experience and now we are prepared to make these talents available to anyone who might feel the need for a little help at times in doing their programming. We are prepared to do this at a very reasonable cost, tailoring the service to fit the needs of those who subscribe.

The programming tips and ideas to be found in the following pages have been prepared from ideas and suggestions received from people who have contacted us over the past six months. The methods and contents of our regular issues in the months to come will grow, change, and be improved according to feedback, comments, and requests from subscribers.

Graphics . . .

WHITE AS A SHEET

By Ed Thorne

On page one I mentioned the Level I program that "whitens" the screen. Now this little exercise we're going to go through will not do anything more than demonstrate some of the capabilities of the Level II computer over the Level I. But just by running through these exercises you will gain some knowledge that will help you in your future programming. At any rate, try this. In case you have forgotten, the Level I program I spoke of is as follows:

To whiten the screen from left to right and from top to bottom

```
PROGRAM A
10 CLS:FOR Y=0 TO 47
20 FOR X=0 TO 127
30 SET(X, Y):NEXT X
40 NEXT Y
50 GOTO 50
. . . AND RUN . . .
```

It only takes a couple of minutes to program and run this. With this program our computer takes about 47 seconds to paint the screen white.

Now on your Level II, try this:

```
PROGRAM B
10 CLS:Z=65
20 FOR Y=0 TO 14
30 FOR X=0 TO 63
40 PRINT CHR$(Z);
50 NEXT X
60 Z=Z+1
70 NEXT Y
80 GOTO 80
. . . AND RUN . . .
```

Now your screen should have filled up in about 8 seconds with a row of "A's" followed by a row of "B's", then "C's", and so on down to a row of "O's" at the end.

I'll explain all that in just a minute. But for the sake of the experiment let's continue on a little further.

```
PROGRAM C
CHANGE LINE 10 TO . . .
10 Z=191
TYPE IN . . . 60 . . . AND ENTER (ELIMINATES LINE 60)
. . . AND RUN . . .
```

Your screen should have been painted white in about 8 seconds. I'll also explain this in a minute but first we'll do one more little program.

```
PROGRAM D
10 CLEAR 63
20 CLS:FOR Y=0 TO 960 STEP 64
30 PRINT@ Y, STRING$(63,191);
40 NEXT Y
50 GOTO 50
. . . AND RUN . . .
```

Now the screen is painted white in less than two seconds! This is about 23 times faster than Level I and about 4 times as fast as in program C.

It is also possible to whiten the screen almost instantaneously by using Assembly Language. That will be covered in a future issue when we tackle Assembly Language.

Now for an explanation of all of the above. Since I am writing for people with different levels of experience in programming, I will sometimes have to ask some of you who already understand what I am explaining to please bear with me. The last thing I want to do is leave someone behind and lost. At times I will get down to an over-simple explanation and work up from there. This is one of those times.

More than one person has called computers, "Nothing more than a super-fast adding machine." That's because a computer — right down at its lowest level — just adds and subtracts numbers. Then how can it keep track of letters and punctuation? It's all done with numbers! If you type in a letter "A", for example, the electronic impulses of the "A" key are really telling the computer, "Hey, that's a 65", or if it's a letter "B", "That's a 66", and so on. Same thing with punctuation.

To make things a little easier, a lot of computer manufacturers got together and decided that it would be better for all concerned if they used the same numbers for the same thing. So they made a standard list of numbers to represent all of the characters and symbols used in computers and they called it the "AMERICAN STANDARD CODE for INFORMATION INTERCHANGE". For short everyone calls it "ASCII", usually pronounced "ASKEY". The whole list of these ASCII numbers is in the Level II manual on pages C/1 and C/2.

(Continued on next page)

WHITE AS A SHEET *(Continued from page 2)*

Now why did I go through all that and what do you care anyway? Well, in programs B, C, and D above we used some of these ASCII numbers to tell the computer what we wanted it to do. It made the programming a lot shorter. On this page you will find drawings of 63 different graphics symbols (usually called blocks) that you can tell the computer to print. With these graphics blocks you can make fancy borders, patterns, or draw pictures. You don't see them on any of the keys because the *only* way you can tell the computer to print them is by using their ASCII numbers. We will be using these graphics blocks a lot in our programs.

Now that I've gotten all that out of my system I can explain programs B, C, and D. If anyone wants a simple explanation of program A please write and say so. I'll be happy to oblige but you will probably understand that after reading the rest of this. Here's a line by line explanation of program B.

PROGRAM B

Line 1Ø. [CLS:Z=65] We clear the screen of anything on it before starting, always a good idea. Next we assigned the ASCII code number for letter "A" to Z (Z=65). It doesn't have to be Z, we could have used X or Y or some other letter, but Z was not assigned to anything elsewhere so we used that.

Line 2Ø. [FOR Y=Ø TO 14] The beginning of a loop. Once again, we didn't have to use Y. But we often use X for horizontal line information and Y for vertical line information. This line is telling the computer we are going to have 15 lines so we used Y.

Line 3Ø. [FOR X=Ø TO 63] The beginning of another loop. Since the first loop is not "closed" yet this loop is inside it or "nested".

Line 4Ø. [PRINT CHR$(Z);] CHR$ is a level II function that has as its only task the job of converting ASCII numbers to the character or symbol they represent. If you type .. PRINT CHR$(65) ... and ENTER ... the computer will print a letter A. If you type ... PRINT CHR$(191) ... and ENTER ... The computer will print Graphics Block 191, which consists of a white rectangle two graphics points wide and three graphics points high (see diagrams of graphics blocks on this page). By programming [PRINT CHR$(Z)] we tell it to look at Z. It sees we have assigned the number 65 to Z and prints the ASCII equivalent of 65, the letter A.

Line 5Ø. [NEXT X] Completes the X loop and starts another one going until all 63 have been done. This prints the line of "A's" across the screen.

Line 6Ø. [Z=Z+1] You can see what this does. It changes the number we had assigned to Z and increases it by 1 so it is now 66, the ASCII code for the letter B.

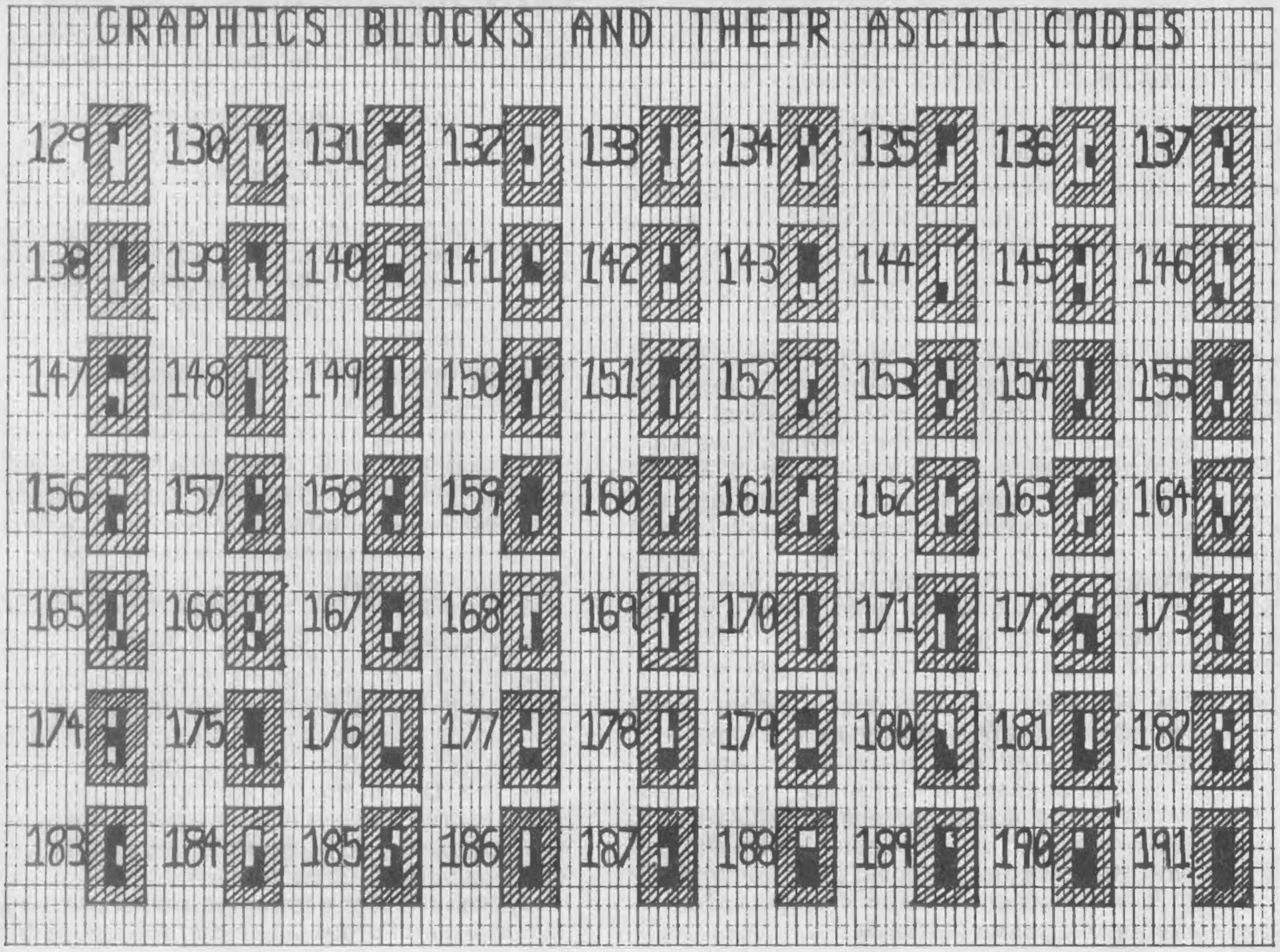
Line 7Ø. [NEXT Y] Completes the Y loop and starts another one going. Then another X loop is started all over again and a line of "B's" is printed on the screen. This looping continues until all 15 lines programmed in the Y loop have been done, then the computer drops to . . .

Line 8Ø. [GOTO 8Ø] This just keeps the computer scrambling back and forth in line 8Ø until the "BREAK" key is pressed. If we didn't have line 8Ø the program would end and "READY<__" would be printed in the last two lines of the screen. This would disrupt the display before you had time to look at it.

PROGRAM C

Program C uses basically the same program that Program B does. We change line 10 to (Z=191) since this program is

(Continued on page 4)



By Mike Kelsey

Building a Program
RECORDS ON DISPLAY

Program by Marvin Mah

Graphs are a versatile and handy method for showing information that might be meaningless on paper. Displaying information on a graph gives it a visual impact and we can see what the numbers mean instead of trying to interpret figures alone.

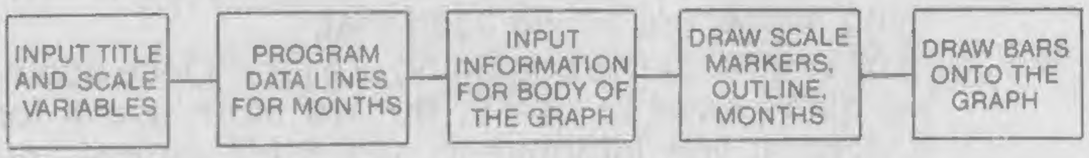
Graphs come in many shapes and forms — point graphs, line graphs, bar graphs, area graphs, to name a few. But before you can see the information you have developed, you first have to build a basic graph program to put on the screen.

For this sample issue, we will show primarily how to build a graph program and get it ready. Several options available to you are described briefly and then one type of graph program is developed. In a regular issue we will develop a number of different ways to display your facts and figures on this or similar types of graphs. Your choice, do it your way! We will go over each line of a program in detail if that part of programming has not been covered before in these pages. We describe each method and function once and after that refer you to where it has been described before.

Let's suppose that you have decided you need a graph to show any of several aspects of business or personal accomplishments. You might want to show business income, expenses, budgets, biorythms, there are hundreds of different reasons for needing a graph. You want to be able to feed the information to the computer and have it draw a graph so you can visualize what the information looks like.

The first step is, of course, a little planning. The scale, time period, and method of showing the information all have to be decided upon and programmed in their proper place. For an example, let's say you want a variable scale so you can put different material into the program and make different graphs for several things. You want to show this information on a monthly basis over a period of a year in a bar graph form.

This program will need the following basic parts, or sub-programs: A basic graph outline; means for putting a title at the top; scale and scale markers for ease of reading; monthly labels; and sub-programs to feed the information for each different graph into the program. If you put all of these program parts into boxes on a piece of paper, the overall programming task can be seen clearly. This first step might look like this:



What you have just done here is to draw a simple flowchart. Flowcharting is another complete subject that will be covered in future issues. It is mentioned here to identify it.

The next step is to tackle these one at a time, programming each part separately then combining the whole thing into a working program. We can take these in the order they appear in the flowchart above and do one at a time.

(Continued on next page)

WHITE AS A SHEET *(Continued from page 3)*

going to print out graphics blocks to whiten the screen and 191 is the graphics block that is all white. Because we want to continue printing this same block throughout the whole program we also have to remove line 6Ø (Z=Z+1) so the number assigned to Z will not change.

PROGRAM D

Program D uses another Level II function, "STRING$(__,__)" (I call it 'STRINGSTRING'). This will accept either the actual symbol itself OR the ASCII code number for a symbol. It also allows us to designate how many of these symbols we want printed. The first number in parenthesis after STRING$ designates HOW MANY of the symbols we want and the second number designates WHICH symbol we want. To print the actual letter we would have to enclose it in quotes. Example: STRING$(5,"Z") will print the letter Z five times. STRING$(63,Z) will print the symbol for whatever ASCII code number is assigned to Z 63 times.

Line 1Ø. [CLEAR 63] When we turn the computer on it automatically clears 50 bytes for use in strings. If we are not going to use more than 50 bytes in any string then it is not necessary to CLEAR any space. (one byte = 1¼ letters, approximately). In PROGRAM C we were only able to use one character in a string at one time. Here, we are assigning 63 characters to a string. It is necessary to clear enough space for them or we will get an OS ERROR.

Line 2Ø. [CLS:FOR Y=Ø TO 96Ø STEP 64] Using STRING$ in this program we will be printing 63 characters in one sweep so we only need one loop to limit the number of lines printed. We want to begin each line at the left edge of the screen so we program (Ø to 96Ø STEP 64). The first line starts at 0, (top line, left edge of the screen) and sweeps across the screen. The next Y steps 64 places which just happens to be the second line, left edge of the screen. And so on until the screen is full of the character we have designated.

Line 3Ø. [PRINT@ Y, STRING$ (63,191)] Let's take this in two parts. PRINT@ Y. As explained above, Y is at Ø and printing starts there. STRING$(63,191) prints Graphics Block 191 63 times which "whites out" the first line across in one sweep.

Line 4Ø. [NEXT Y] Starts the Y loop over again and continues until the screen is completely white.

Line 5Ø. [GOTO 50] Locks things up so we can see the display.

You may have noticed that in program D I only programmed to print 63 characters when I could have printed 64. Reason: If we print the full 64 characters, when the last line is completed (and full of characters), the computer automatically moves everything up one line and is ready to print the next line. But all the Y's are used up and there is nothing more to print. We would have a blank line at the bottom and the first line would be gone. So we print only 63 characters and this does not happen.

We will use all this to make borders, patterns, and pictures in subsequent programs and future issues of the Letter.

Building a Program *(Continued from page 4)*

I am going to go counter-current to established practice and call these instructions. Just about everyone else calls them statements.The term "statement" and my use of the term "instruction" should be considered synonymous or having the same meaning.

There are several calculations within the program lines. I have devoted a whole section just to a discussion of these calculations. For an explanation of the calculations contained in the lines below you are referred to that section.

Any instruction that has a colon (:) after it is a complete instruction in itself. We could use a separate line for each of these instructions. By combining them two or more to a line we save memory space.

INPUT TITLE AND SCALE VARIABLES Referring to the complete program, this is done by lines 4Ø and 5Ø.

LINE 3Ø CLEAR 2ØØ:DIM D(12),M$(12) The CLEAR function was discussed in the explanation of PROGRAM D under GRAPHICS. In running this program with various titles, different scales, and other test data, we occasionally got an OS ERROR. For this reason the CLEAR was increased to 2ØØ as being more than enough to cover whatever variables might be used. CLEAR will be covered in more detail in another issue.

DIM D(12),M$(12) When the computer is turned on, it automatically sets up space for 11 subscripts in any array (Ø-1Ø). Here, we are using 1-12 and must DIMension them first. Otherwise we would get a BS ERROR.

LINE 4Ø CLS:INPUT"WHAT IS THE TITLE OF THE GRAPH";T$ Allows you to type in any name for the graph and assigns it to T$.

L=LEN(T$): LEN(?$) is the Level II function that finds how many characters are in any string. Here it is used to count the characters in the title, assigning this number to L.

T=INT((5Ø-L)/2)+76 This is a calculation that determines where the title will be printed so it will be centered. (See section on calculations, below).

LINE 5Ø INPUT"THE MINIMUM VALUE OF THE SCALE IS Ø. WHAT IS THE MAXIMUM VALUE";H: This allows you to type in any maximum graph scale you wish (within reason, of course).

F=H/10 Another calculation, this one computes a "scale factor" that is used later to print the scale down the left side of the graph.

PROGRAM DATA LINES FOR MONTHS This data is necessary before any more input can be programmed. This is done in line 6Ø and here is another option you have when making the program. The year doesn't have to start with January. You can start with any month you choose in the data line, just remember that the computer will start reading with the first month you've programmed. It doesn't have to be months. You could use years (two numbers) or any labels that do not exceed three characters. For this program there must be 12 labels. For more, or less, other parts of the program must be changed slightly. A number here and a number there.

LINE 7Ø FOR A=1 TO 12:READ M$(A):NEXT A This is a complete loop in one line. It reads line 6Ø and puts all the months (or other labels you have put into line 6Ø) into a string array, M$(1) through M$ (12).

INPUT THE INFORMATION FOR THE BODY OF THE GRAPH The last part of the INPUT section, this is where you can supply the information you want to see on the graph. It is all done in line 8Ø, another complete loop in one line.

LINE 8Ø FOR A=1 TO 12:PRINT"WHAT IS THE DATA FOR "M$(A);: INPUT D(A):NEXT A Here the computer asks us for data, pulls the first month out of array M$, and waits until we type in the data and press ENTER. Then it asks the question again and pulls out the next month and so on. Each time we press ENTER, the computer stores the data we have just typed into another array, D(1) through D(12), where it will be stored until needed later. The loop counter A is used here to provide the subscript for both arrays. As soon as the last piece of data has been ENTERed (A=12), the computer races on to next line.

DRAW THE SCALE MARKERS, OUTLINE, MONTHS Since all the variables have been entered, we are now ready to draw the graph and do something with it. This is the fun part — seeing the graph take place before your eyes. Let's follow what happens.

LINE 9Ø CLS:FOR S=Ø TO 1Ø:PRINT@ 192+S*64,F*(1Ø-S);: If I print the whole line out here I'll scare you. So let's analyze one instruction at a time. We start out by setting up a loop, FOR S=Ø TO 1Ø. Then we give a print instruction. This print instruction turns out to be two more calculations. PRINT@ 192+S*64, is telling the computer *where* to print the first scale marker. We could print out 1Ø scale markers by using 1Ø different lines. This way we do it all in only one line.

(Continued on page 6)

GRAPH PROGRAM LISTING

```
1Ø  REM GRAPH PROGRAM . . . WRITTEN BY
    MARVIN MAH, FEB 1979
3Ø  CLEAR 2ØØ: DIM D(12),M$(12)
4Ø  CLS:INPUT"WHAT IS THE TITLE OF THE
    GRAPH";T$:L=LEN(T$):T=INT((5Ø-L)/2)+76
5Ø  INPUT"THE MINIMUM VALUE OF THE SCALE
    IS Ø.WHAT IS THE MAXIMUM VALUE";H:F=H/1Ø
6Ø  DATA JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,
    SEP,OCT,NOV,DEC
7Ø  FOR A=1 TO 12:READ M$(A):NEXT A
8Ø  FOR A=1 TO 12:PRINT"WHAT IS THE DATA
    FOR "M$(A);:INPUT D(A):NEXT A
9Ø  CLS:FOR S=Ø to 1Ø:PRINT@ 192+S*64,
    F*(1Ø-S);:H$=CHR$(136):PRINTTAB(9);
    H$;TAB(11);CHR$(132);TAB(24);H$;TAB(36);
    H$;TAB(48);H$;TAB(6Ø);H$:NEXT S
1ØØ PRINT@ 1Ø,CHR$(188):PRINT@ 63,CHR$(188)
11Ø FOR G=1 TO 13:G$=CHR$(191):PRINT@
    1Ø+64*G,G$;:PRINT@ 63+64*G,G$;:NEXT G
12Ø G$=STRING$(52,14Ø):PRINT@  11,G$;:PRINT@
    139,G$;:PRINT@ 843,STRING$(52,176)
13Ø PRINT@ T,T$;
14Ø FOR A=1 TO 12:PRINT@ 9Ø5+A*4," ";M$(A);:
    NEXT A
15Ø X=31:FOR A=1 TO 12:D=41-(D(A)/F)*3
16Ø FOR Y=41 TO D STEP-1:SET (X,Y):SET(X+1,Y):
    NEXT Y
17Ø X=X+8:NEXT A

2ØØ GOTO 2ØØ
```

Building A Program *(Continued from page 5)*

F*(1Ø-S);: Now we're telling it *what* to print at 192. In line 5Ø we calculated what F was. It's maximum scale value divided by 1Ø, and remains the same through each individual graph. If your maximum scale value is 1ØØ then F=1Ø. If it is 1ØØØ then F=1ØØ, etc. In this case F=1Ø. 1Ø*1Ø=1ØØ and 1ØØ is printed at 192 for the first, or top, scale number.

H$=CHR$(136):PRINTTAB (9);H$;TAB(11);CHR$(132); This prints a little "tick" or marker on either side of where the left vertical graph border is going to be in about two seconds. This is to aid the eye in reading the graph when finished. To refresh your mind, if necessary, refer back to page 3 under GRAPHICS. In the explanation for program B, line 4Ø, we discussed the CHR$ function. By assigning CHR$(136) to H$ we saved memory space. Each time we print H$ the computer prints CHR$(136).

TAB(24);H$;TAB(36);H$;TAB(48);H$;TAB(6Ø);H$:NEXT S All this does is to TAB across the screen (just like a typewriter) and put four more markers across on the same line for the same reason. If you used Jan as your first month then these markers will separate the quarters of the year. NEXT S sends the computer back, changes S to 1, and goes through the line again. This time the calculations produce the next scale number and another row of markers across the screen.

LINE 1ØØ PRINT@ 1Ø,CHR$(188):PRINT@ 63,CHR$(188); This prints the upper right and left corners of the graph outline.

LINE 11Ø FOR G=1 TO 13:G$=CHR$(191):PRINT@ 10+64*G,G$;: PRINT@ 63+64*G,G$;:NEXT G This line prints the left and right vertical lines of the border. To do all this in one line we use another loop and two more calculations to get things lined up properly.

LINE 12Ø G$=STRING$(52,14Ø):PRINT@ 11,G$;:PRINT@ 139,G$;: PRINT@ 843,STRING$(52,176) This line completes the graph outline by printing the horizontal lines.

G$=STRING$(52,140): Assigns the STRING$ function to G$, saves us having to type in the whole series each time. Saves time, saves memory.

PRINT@ 11,G$;:PRINT@ 139,G$;: Should be self-explanatory. By assigning STRING$ to G$, every time we use PRINT@ ??,G$, we get the whole STRING$ function, in this case a horizontal line, zap!

PRINT@ 843,STRING$(52,176). In this case, we wanted to use a different graphics block for the bottom line, so had to spell it all out.

LINE 13Ø PRINT@ T,T$ Prints the title, T$, at T, the location calculated in line 4Ø.

LINE 14Ø FOR A=1 TO 12:PRINT@ 905+A*4," ";M$(A);:NEXT A This prints, at their proper locations, the month labels across the bottom of the graph.

O.K. Now we have done everything except draw the bars to show the information we did all this for in the first place. The computer has all it needs to know stored inside except how to put this information on the screen. Now we will program those instructions.

DRAW BARS ONTO THE GRAPH

All right, now everyone who understands what the X and Y coordinates are on the TRS-8Ø raise your hand. Hmm . . . almost everybody. O.K., what I'll do for the few who aren't quite sure is explain coordinates down below, just following the section on Calculations. If you know, just keep on reading. If you're not sure, check yourself out down below and then come on back.

LINE 15Ø X=31:FOR A=1 TO 12:D=41-(D(A)/F)*3 X=31 sets the X coordinate for printing the first part of the first bar. FOR A=1 TO 12: begins a loop to print bars for the 12 months.

D=41-(D(A)/F*3) Another calculation. This takes the data you input back in line 8Ø and calculates how high the bar will be for each month before it starts printing.

LINE 16Ø FOR Y=41 TO D STEP-1:SET(X,Y): Now we start a second, or nested, loop to actually print the bar. The Y becomes the Y coordinate for the bar and its limits are established by the calculation made in line 15Ø. SET(X,Y) turns on the first graphics point of the bar.

SET(X+1,Y):NEXT Y Setting just one graphics point for a bar makes it look pretty skinny. We decide that a wider bar would look much better. SET(X+1,Y) doubles the width of the bar. NEXT Y starts the loop over again, the Y coordinate is reduced by 1 (STEP-1) and the bar continues up until it reaches its limit, D.

LINE17Ø X=X+8:NEXT A We finished with Jan (or whatever else you programmed) in LINE 16Ø and X=X+8 moves the X coordinate to a position over FEB. NEXT A processes and prints the next bar and so on until all have been drawn and the graph is complete.

There is a drawing on page 7 of what this graph looks like when run with some test data. The test data is: Jan 1Ø; Feb 2Ø; Mar 3Ø; Apr 4Ø; May 5Ø; Jun 6Ø; Jul 7Ø; Aug 8Ø; Sep 9Ø; Oct 1ØØ; Nov 9Ø; Dec 8Ø.

CALCULATIONS

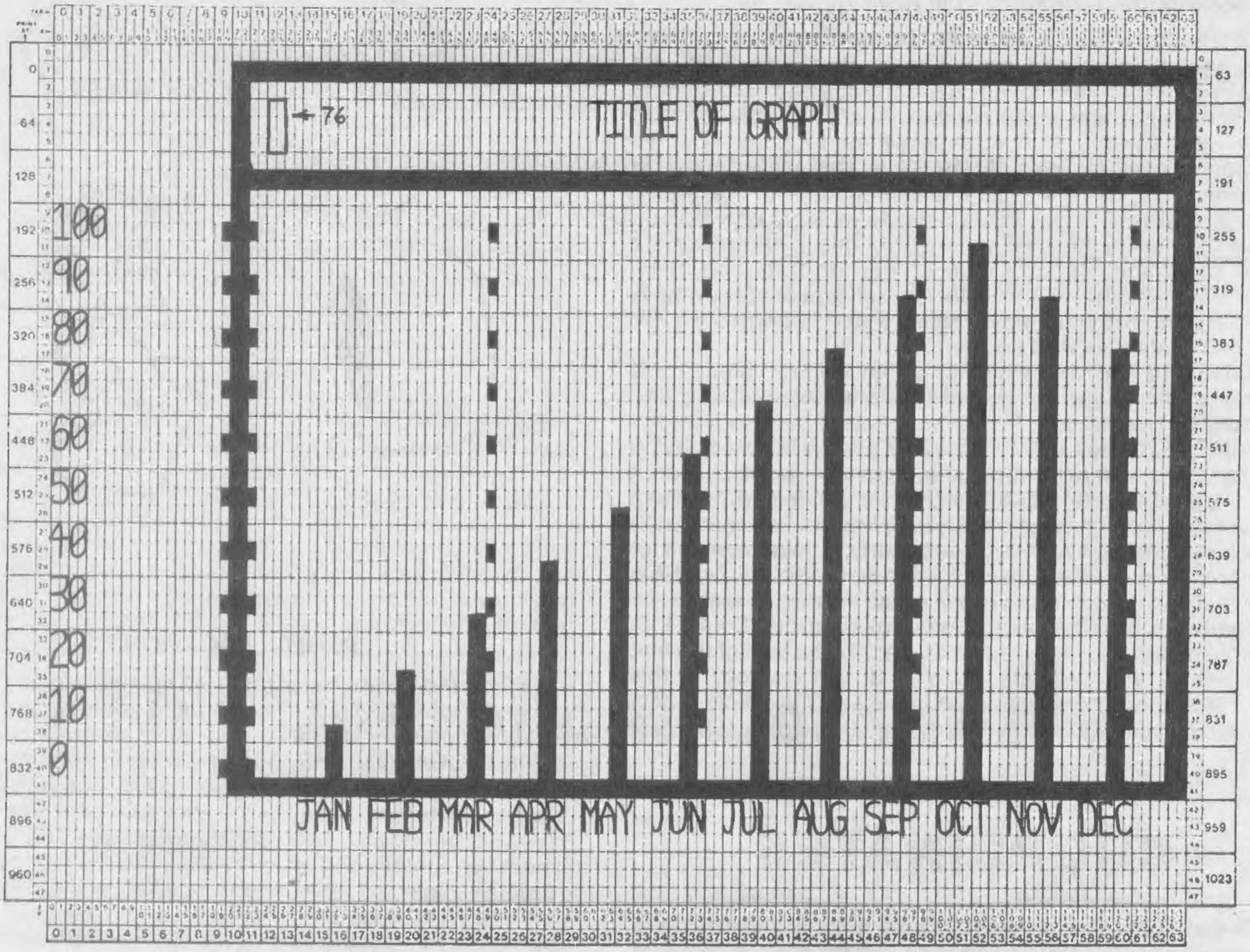
If you understood the calculations in the program lines, you may want to skip over this section.

In 1978 I wrote a booklet called "HOW TO CHOOSE *YOUR* CALCULATOR". Some of the material in this section is taken from that booklet. (24 pages, available from the Mail Mart, $2 postpaid). There are 6 different kinds (including variations) of logic to be found in present day calculators.

> LOGIC: The fundamental principles and the connections of circuits for arithmetical computation in a computer; also: the circuits themselves . . . Webster's New Collegiate Dictionary, 1973.

We need not concern ourselves with all of these different ways of approaching a problem since the TRS-80 uses only one of them. And that logic is Algebraic Logic with Hierarchy and Parenthesis. (My description of it). Now all that big fancy name means is that the computer itself sorts out all of the multiplication and division from the addition and subtraction and does them in a certain order. It also has parenthesis so you can sort out some of it yourself in case you don't like the way the computer does it.

(Countinued on next page)



By Mike Kelsey

Calculations *(Continued from page 6)*

On page 1/6 of the Level II Reference Manual you will find a list called "Order of Operations". This list tells you what the computer will do first, second and so on. When programming a calculation all this has to be kept in mind.

As you know, we can assign numbers to letters and the computer will keep them stored until we need them. Then, when they are needed, we just need to put that letter in the program line and the computer digs up the information stored or assigned to that letter and uses it just as if we had programmed the actual number itself.

In the previous program, and any other program for that matter, we used many of these assigned numbers in calculations to tell the computer *where* to print and *what* to print. Just looking at the letters in a calculation can be confusing. To be able to see just what we are computing it is necessary to substitute the actual numbers for the letters and make the calculation that way, keeping in mind the Order of Operations.

There is a little test you can make to prove to yourself that the computer really does sort out and do some things before others. Try this line: 1Ø PRINT 7+6*5/(4-2). Run this and you will get an answer of 22. (If you have a calculator handy, try it on that. If the calculator does not have parenthesis keys then just skip them. You will get an answer of: −5.5, 5.5, 12.5, 14.25, 22, 32.5, or 2 depending on the type of logic in individual calculators!) If you did this in the order it is written you would get this: 7+6 (=13) *5 (=65) /(4-2) (=32.5).) So the TRS-80 really has sorted out the multiplication and division and done that first.

To get back to the graph program, we can go through some of the calculations just in case anyone got lost. Take line 4Ø. It contains the calculation (T=INT((50-L)/2)+76). In this graph program there are 5Ø spaces alloted for the title. The first PRINT@ location for the title is 76. This calculation subtracts L, the number of characters in the title, from the 5Ø spaces alloted, which leaves the number of blank spaces there will be. It then divides the number of blank spaces by 2 and adds the result to 76 which is the first PRINT@ location for the title. The final result of all this is the PRINT@ location where this particular title will begin to be printed. The calculation allows you to have a title of any number of characters up to 5Ø and will center any of them. In this case, we are not ready to print the title yet so the number is assigned to T and stored until needed.

In LINE 5Ø the calculation is (F=H/1Ø). H is the maximum scale value you typed in. We have room on the graph for 1Ø graduations of scale to be listed down the left side. Dividing the maximum by 1Ø gives us a "scale factor" or fraction to compute what each scale graduation will be. This is assigned to F for later use.

LINE 9Ø has a calculation that uses the loop counter, (S=Ø TO 1Ø:). The loop counter starts out at Ø because the first time aound we want to print right at 192. Substituting Ø

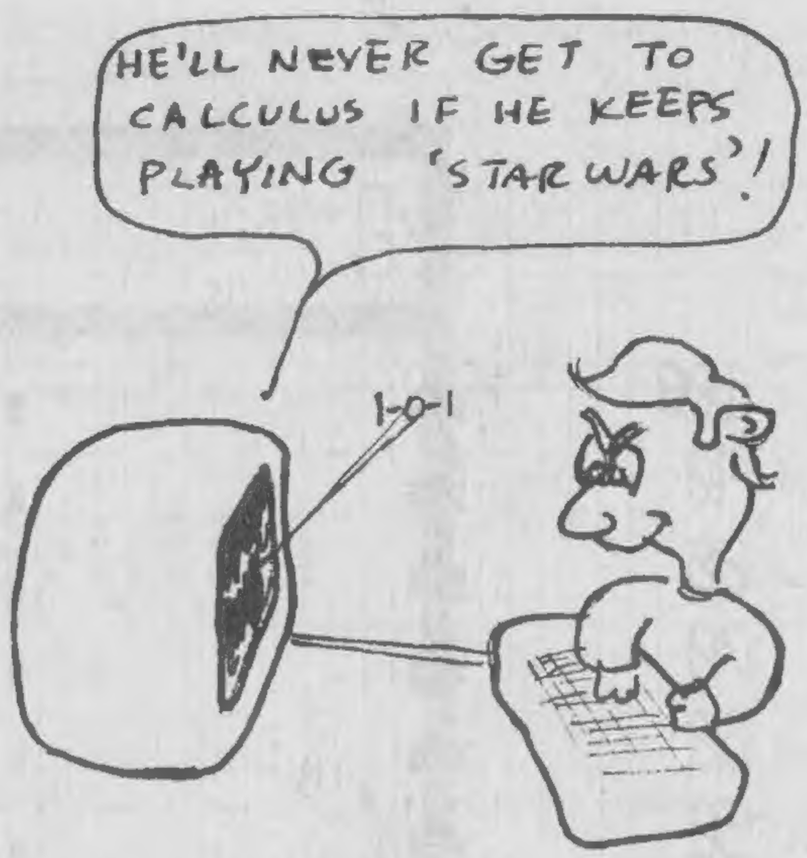
(Continued on page 8)

Calculations *(Continued from page 7)*

for S we have Ø*64=Ø. 192+Ø=192, our first print location. The next time around on the loop, S=1. 1*64=64 and 192+64 just happens to put us exactly one line below where we printed our first scale number. And so on, lickety split, down the screen. That told us *where* to print, now we compute *what* to print. [F*(1Ø-S)] F, the scale factor, or fraction, multiplies (1Ø-S). First time around, S=Ø. We divided by 1Ø in line 5Ø to get F and now we multiply it by 1Ø and have the maximum scale value back. This is printed on the first scale line. Second time around on this loop, S=1. 10-1=9, multiplied by the scale factor F again, now we have 9/1Ø of the maximum scale value which is printed on the second scale line. And so on.

LINE 11Ø has two calculations that are just variations of the first one in line 9Ø and also arrange to print one line exactly below the last.

LINE 15Ø (D=41-(D(A)/F)*3) That's a lulu! Here we're also using the number of the loop counter to provide the subscript for the array. Let's make like the computer and go down inside the parenthesis first. (D(A)/F) For the first month the loop counter is at 1 and D(A) becomes D(1). In our test run for the graph we used a data 1Ø for Jan so D(1) is 1Ø. F is 1Ø in our test run. Dividing D(1) by F comes out to 1, inside the parenthesis. Now the calculations looks like this: (D=41-1*3) Multiplication first, 1*3=3 and 41-3=38. The limit of the first bar will be Y coordinate 38. The 3, last number in this calculation, is a constant since there are 3 graphics points for each scale graduation. This gives our graph a 3 percent error.



COORDINATES

Probably what has confused some people when trying to figure out coordinates and printing locations is the fact that there are three separate systems of numbering different locations on the screen of the video monitor. Let's not knock that because it does give us more alternatives when programming. The three systems of numbering are: X and Y coordinates; PRINT@ locations; and PRINTTAB locations. Each are numbered in a different way.

X AND Y COORDINATES

Refer to the Video Display Worksheet on page 106 of the Level I manual or page E/1 of the Level II Reference Manual. There are 6144 of the small rectangles on that sheet that represent locations that can be turned on (or whitened, or lighted). The X and Y coordinates are just locations or addresses to specify which one of these 6144 rectangles we are referring to. I will call them GRAPHICS POINTS. We can light them or turn them on by using the SET command, along with the horizontal address (X coordinate) and the vertical address (Y coordinate) for that individual GRAPHICS POINT. They are turned off by using the RESET command along with the same addresses or coordinates. These coordinates are shown along the top and sides of the Worksheet, Ø-127 for the X coordinates and Ø-47 for the Y coordinates.

The computer has to have the actual numbers before it can turn these points on or off and this is often done by assigning the numbers of these coordinates to letters, usually X and Y. They don't have to be X and Y, you can use A and B, A and Z or other combinations. If we used A and B, for example, we would then also have to SET(A,B). Since they are known as X and Y coordinates, X and Y are usually used. These are the smallest areas of the screen that we can light up at any time. We used them to draw the bars in the graph program.

PRINT@ LOCATIONS

When the computer prints a letter, number, punctuation, or graphics BLOCK, the space used takes up six of these graphics points. I will call this block of six graphics points a PRINT LOCATION. PRINTing one letter or number uses one Print Location. There are 1Ø24 Print Locations on the screen. Referring again to the Video Display Sheet, these Print Locations are outlined by heavier lines. There are 64 of them horizontally and 16 vertically. They are numbered from Ø, at the upper left of the screen, to 1Ø23, at the lower right of the screen. By programming PRINT@ and using one of these numbers we tell the computer to hop right over to that location and print. It does not erase anything except what is in that one location.

PRINTTAB LOCATIONS

While the PRINT@ command will run the computer all over the screen to print something, the PRINTTAB only works on the line that is currently being printed and only to the right of the cursor. Since it only works on one line at a time, the PRINTTAB numbers are the same for all lines. They run from Ø at the left, to 63 at the right side of the screen. We can start a line at a PRINT@ location, then TAB it to any other location to the right in the same line. In doing this it works almost the same as a typewriter.

IN CONCLUSION . . .

As this is being written, we have a number of things in the works for the issues to come. Whether we use all of these things or not depends largely upon what you, as a subscriber, put down on the enclosed Subscriber Questionnaire. The only way we can be really helpful and effective is to publish the type of article desired by those who subscribe to our service. I urge you to fill out the Questionnaire and return it in the postage-paid envelope as soon as possible. A summary of Questionnaire replies will be published from time to time.

If you have information or programs you would like to share with others, send them in for our appraisal.

If you would like to write an article to be published in a future edition, send us an outline of what you have in mind.

If you would like to subscribe to this newsletter and do not have an order card, just send your check for $24.00 (for a one-year subscription) to: The Mail Mart PO Box 11102 San Francisco, CA 94101 along with your name and address.